3D Reconstruction of Live Chickens in Poultry Houses

(Research Option – Thesis)

Aneri Muni

Faculty Advisor: Dr. Fumin Zhang

Project Manager: Colin Usher

Food Processing Technology Division, Georgia Tech Research Institute

Georgia Institute of Technology

Faculty member 1

X

Dr. Fumin Zhang

Faculty Advisor

Faculty member 2

X

Colin Usher

Project Manager

3D Reconstruction of Live Chickens in Poultry Houses

(Research Option – Thesis)

Aneri Muni
Faculty Advisor: Dr. Fumin Zhang
Project Manager: Colin Usher
Food Processing Technology Division, Georgia Tech Research Institute
Georgia Institute of Technology

Abstract

Poultry houses require daily monitoring to ensure animal health and proper house operation. One task involves observing the average growth rate of the house to adjust the daily feed. In addition to being labor intensive and time consuming, it is difficult for the farm owners to find consistent labor to fill these jobs. This project looks at the possibility of estimating the weight of a chicken based on the volume of the chicken as captured by a 3D model. We present a system capable of reconstructing dynamic scenes, i.e. chickens in a poultry house, by fusing together depth scans captured using a Microsoft Kinect. Like *DynamicFusion*, our approach involves discretizing the live depth frame into nodes and estimating individual 6D transformations before fusing them together to reconstruct scene geometry. This approach doesn't use any prior model of template, making it applicable to a wide range of dynamic objects and scenes.

1. Introduction

Poultry farmers take a walk through chicken houses couple of times a day to pick up eggs laid on the ground, to check for sick or injured chickens and to check the feed and water supply. Farmers may carry disease agents into the house, on their hands, clothes or foot ware. On the other hand, they may contract diseases from birds and their waste products. There is a need for a non-invasive techniques to automate some of the farmers' tasks, in both the grow-out houses and the processing plants. This project looks at the possibility of estimating the weight of a chicken based on the volume of the chicken as captured in a 3D model.

There is evidence of a correlation between the weight and volume of a chicken [1, 2]. Current approaches using single dimension measurements (e.g. body length, size of breast) to correlate weight and volume show promise. Estimating volume by fitting spheres or other geometric shapes in 3D scans of chickens fails due to the fact that new born chickens are more rounded but older ones have elliptical bodies. Furthermore, using ICP (Iterative closest point) to align two scans of chickens yields hydra chickens with two heads and tails. Thus we propose to implement and validate an algorithm for generating template free 3D models of live chickens in real-time, using a depth sensor, Microsoft Kinect.

Numerous efforts have been made to develop robust algorithms to generate detailed 3D models of objects, outdoor and indoor scenes. There are several desired properties [3] for surface reconstruction algorithms including incremental updating, range uncertainty representation, gap filling ability and real time and template free reconstruction of dynamically changing scene. Prior work in the field fails to fulfil at least one of these properties. The *KinectFusion* [4] algorithm covers most of the properties but it is developed with the assumption that the observed scene is largely static. *DynamicFusion* [5] generalizes *KinectFusion* to incorporate tracking of non-rigid scenes. It tracks the motion of the sensor while simultaneously reversing the motion of scene to fit to the initial frame, in real-time. *VolumeDeform*, further utilizes the color map as SIFT (Scale Invariant Feature Transform) features to avoid drift [6]. While the use of sparse photometric feature correspondences help to reduce drift and in tracking tangential motions, this project will not employ color for tracking. Our target object, chicken, does not provide color enough variation to use SIFT features as global anchor points.

We propose to create an algorithm, similar to DynamicFusion to autonomously estimate the average weight of birds in a poultry house by reconstructing 3D models for dynamically changing scenes in real-time. The algorithm will reconstruct an implicit surface representation of the tracked object, while jointly optimizing for the scene's rigid and non-rigid motion based on a coarse warping field.

2. Previous work

The overall goal is to the construct 360 degree view models of live chickens and develop a correlation between their mass and volume. This project can be divided into two major subparts. The first part involves developing an algorithm to construct 3 dimensional models of chickens. The next step will be to devise techniques to estimate the bird's volume from the reconstructed model and correlating it with its weight.

2.1 3D Reconstruction

Previous work in the field of surface reconstruction can be broadly divided into two categories: The *offline, template free reconstruction method* was an initial attempt to reconstruct 3D models of objects using depth sensors by collecting depth maps of an object from multiple viewpoints. Then, ICP or similar algorithms were used to patch these range images together, offline. This process took several hours and was not robust in handling occlusions [1, 2]. *The online, template based reconstruction method* involves an initial sparse template used as a starting point and has new depth images being overlaid, depending on the structure of the template.

Izadi, Shahram, et al. suggest a robust algorithm *KinectFusion*, for template free scene reconstruction in real time. This is algorithm is years ahead of its predecessors and produces template free reconstructions in real time. It utilizes novel GPU pipeline implementation which significantly increases the computation rate thus allowing real time reconstruction [3].

One major drawback of the aforementioned techniques is that they all operate under the assumption that the scenes being reconstructed are for the most part, static. Chang and Zwicker suggest a novel technique to model the motion of articulated objects using reduced Deformable model (RDM). Predefined motions for the reconstructed object were defined using constraints and the model used to undo the motion of the body [7]. Most recently, R. Newcombe, et al. builds on *KinectFusion*, proposing an algorithm, *DynamicFusion*, to successfully achieve template free reconstruction of dynamically changing scenes, in real-time [5]. They use Warp fields to define transformations from live frame into a canonical frame, thus parameterizing the change in structure to undo the scene's motion. Matthais Innmann, et al. introduces VolumeDeform [6], which takes DynamicFusion a step further, by including extracted sparse color features to avoid drifting and enables robust tracking of fast motions. We will mainly focus on the techniques developed for DynamicFusion.

2.2 Estimating Growth Rate

For the second part, numerous attempts have been made by the agricultural industry to develop integrated systems for livestock monitoring [8]. Earlier attempts to monitor the growth rate of pigs were made using images from wide lens camera fitted above feed stations [9, 10]. Each pig had an electronic ear tag which would be recorded along with its image, when it neared the feeding trough. The pigs were then manually weighed and their volume was estimated by the amount of area covered by the pig in the image captured. In an attempt to analyze the change in mass with growth of a chicken, a set up with a depth sensor attached above a weighting scale was placed in poultry house. Generated point clouds of chickens were used to create meshes and the volume to mass correlating was done by fitting largest radius spheres in these meshes [2].

Anders Krogh Mortensen, et al. used computer vision and neural networks to segment out chickens in poultry houses and estimate their volume [11]. He used linear measurements, shape fitting and convex volume from depth sensors to correlate weight and volume.

Previous experiments required a setup with a depth sensor above a weighing scale as shown in **Figure 1.** This setup will only work if the bird decides to jump on the scale and have its data recorded. Thus the average weight volume outcome of the experiment cannot be generalized for the entire house.

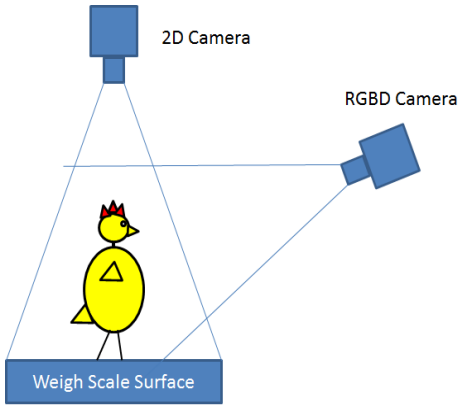


Figure 1. Typical setup for collecting mass-volume data for livestock [2].

We propose to install a depth sensor on an autonomous robot that will reconstruct template free, 360 degree models of chickens in a poultry house and estimate the average weight volume correlation of birds in the house, in real time.

3. Methods and Material

KinectFusion provides 3D object scanning and model creation using a depth sensor, specifically a Microsoft Kinect sensor. The user can simultaneously see and interact with, a detailed 3D model of the scene. *DynamicFusion* builds on the concept of *KinectFusion* reconstructing scene geometry whilst simultaneously estimating a dense volumetric 6D motion field that warps the estimated geometry into a live frame. Like *DynamicFusion*, our system produces increasingly denoised, detailed, and complete reconstructions as more measurements are fused, and displays the updated model in real time. Our method for 3D reconstruction is a variant of the *DynamicFusion* technique for undoing motions in dynamic scenes.

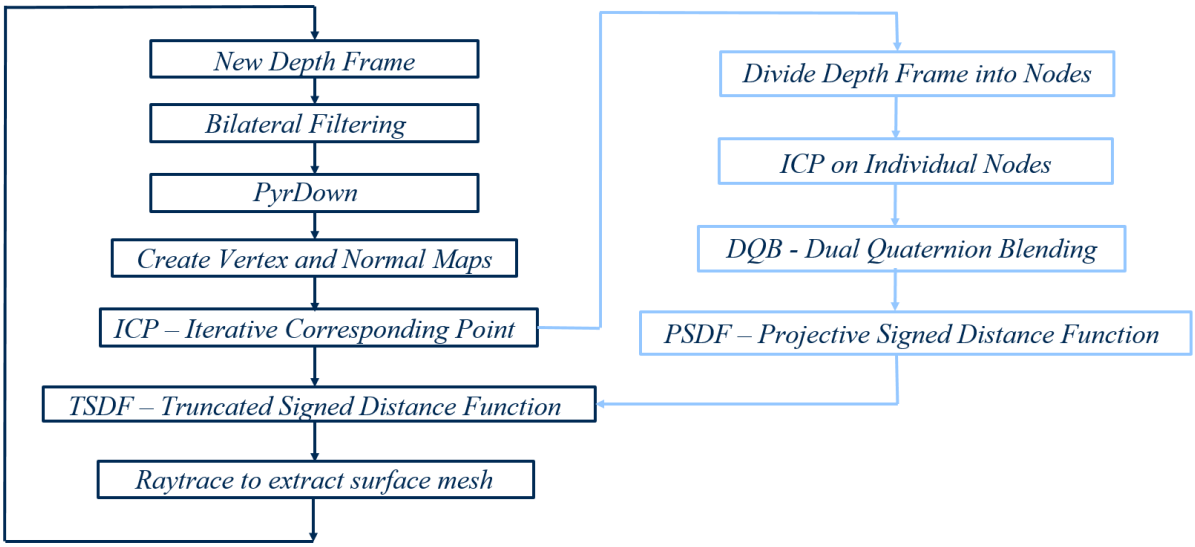


Figure 2. Comparison between KinectFusion (dark blue only) and DynamicFusion (light blue + dark blue) workflow.

3.1. Hardware Requirement

The project is written in C++ with various parts parallelized using CUDA, a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). The project uses an Alienware laptop with Intel Core i7 processor and NVIDIA GeForce GTX 1070 GPU with 16GB Memory and 1TB Hard Drive with 128GB Solid State Drive. Furthermore, the project is built to grab depth frames from a Microsoft Kinect 2.

3.2. Software

Both *KinectFusion* and *DynamicFusion* are not open source. This project utilizes an open-source implementation of *KinectFusion*, KinFu *[12]*, as described by Izadi, Shahram, et al, and available as an extension of the Point Cloud Library. Our project builds on this implementation.

3.3. Overall flow

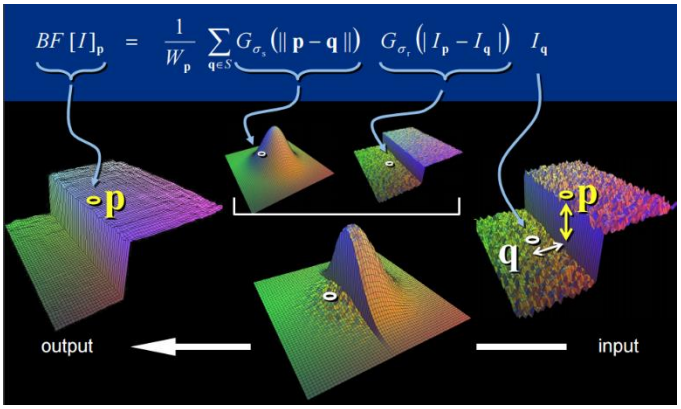
3.3.1. Depth Frame

The Kinect grabber outputs raw depth images. The depth data is the distance, in millimeters, to the nearest object at that particular (x, y) coordinate in the depth sensor's field of view. The project uses a depth image of resolution: 640x480 (the default). It is to be noted that the "Z" value is in the direction of the camera's Z axis, and not to the absolute Z axis of a scene.

Cubic Structure | Depth Map: Nearer is darker | Depth Map: Nearer the Focal Plane is darker

3.3.2. Filtering

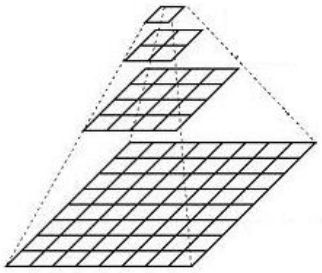
The raw depth data is then passed through a bilateral filter, a technique to smooth images while preserving its edges.



σ_s Spatial extent (size of neighborhood considered), σ_r Range (amplitude)

3.3.3. PyrDown

A Gaussian Pyramid is used to down sample the images. Three levels are created, each eliminating every even-number row and column. This process helps to speed up the Iterative Closest Point algorithm.



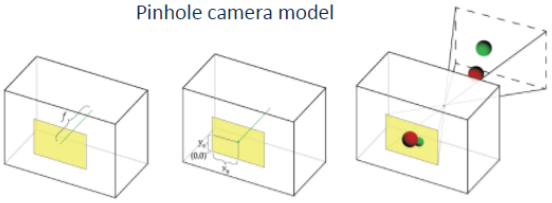
3.3.4. Vertices and Normal maps

The next step converts the raw depth values obtained from the Kinect sensor into vertices and normal maps. This operation is implemented at each of the three levels mentioned in 3.3.3. The Pinhole Camera Model is used to project each pixel from camera frame to world frame.

Pinhole camera model

Camera Intrinsic matrix

$$K = \begin{pmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}$$



"Visibility cone"

f - focal length ($f_x = f_y$)

s - skew (= 0)

(x_o, y_o) - dimension of image plane

K: Transformation between image plane and camera frame

Given that the camera matrix for the Kinect is known, we can use the following equations to obtain the coordinates of the corresponding vertices and it's normal.

Vertex:

$$\mathbf{v}_k = D_k(\mathbf{u})\mathrm{K}^{-1}(u, v, 1)^{\mathrm{T}}$$

Depth of pixel **u** — Intrinsic matrix of Kinect — Location of pixel **u**

where,

u : *(u, v)*

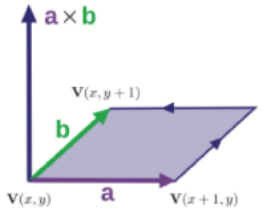
Pixel coordinate

Normal:

$$\mathbf{N}(x, y) = \frac{\mathbf{a} \times \mathbf{b}}{\|\mathbf{a} \times \mathbf{b}\|}$$

where,

a, b *vectors to adjacent vertices*



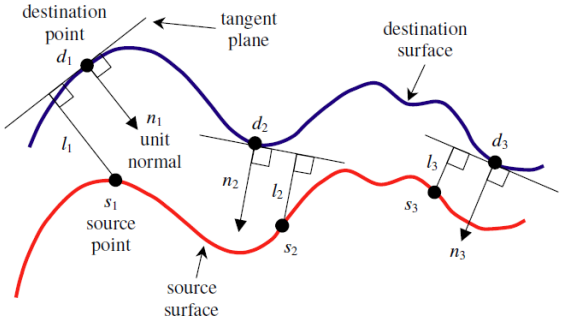
Cross product of vertices

3.3.5. Iterative Closest Point (ICP)

The key concept of the standard ICP algorithm in two steps:

1. Compute correspondences between the two scans.
2. Compute a transformation which minimizes distance between corresponding points.

To compute the correspondences, a Point-to-Plane distance is used. This is done by minimizing the sum of squared distances from each source point to the plane containing the destination point and oriented perpendicular to destination normal.



The second step requires the calculation of a 3D rigid-body transformation, M composed of a rotation matrix R(α, β, γ) and a translation matrix T(t_x, t_y, t_z). M can then be represented as,

$$M = R(\alpha, \beta, \gamma) * T(t_x, t_y, t_z)$$

$$\mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{R}(\alpha, \beta, \gamma) = \mathbf{R}_z(\gamma) \cdot \mathbf{R}_y(\beta) \cdot \mathbf{R}_x(\alpha) = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{aligned} r_{11} &= \cos\gamma\cos\beta, \\ r_{12} &= -\sin\gamma\cos\alpha + \cos\gamma\sin\beta\sin\alpha, \\ r_{13} &= \sin\gamma\sin\alpha + \cos\gamma\sin\beta\cos\alpha, \\ r_{21} &= \sin\gamma\cos\beta, \\ r_{22} &= \cos\gamma\cos\alpha + \sin\gamma\sin\beta\sin\alpha, \\ r_{23} &= -\cos\gamma\sin\alpha + \sin\gamma\sin\beta\cos\alpha, \\ r_{31} &= -\sin\beta, \\ r_{32} &= \cos\beta\sin\alpha, \\ r_{33} &= \cos\beta\cos\alpha. \end{aligned}$$

A linear approximation is made to simplify the above rotational matrix.

When an angle $\Theta \approx 0$, we can use the approximations $\sin\Theta \approx \Theta$ and $\cos\Theta \approx 1$. When $\alpha, \beta, \gamma \approx 0$,

$$\mathbf{R}(\alpha, \beta, \gamma) \approx \begin{pmatrix} 1 & \alpha\beta - \gamma & \alpha\gamma + \beta & 0 \\ \gamma & \alpha\beta\gamma + 1 & \beta\gamma - \alpha & 0 \\ -\beta & \alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\approx \begin{pmatrix} 1 & -\gamma & \beta & 0 \\ \gamma & 1 & -\alpha & 0 \\ -\beta & \alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \hat{\mathbf{R}}(\alpha, \beta, \gamma).$$

It is to be noted that the product of two small numbers gives a smaller number, hence the angle product terms can be approximated to zero. We can now represent the approximate camera to world frame (camera pose) transformation as,

$$\hat{\mathbf{M}} = \mathbf{T}(t_x, t_y, t_z) \cdot \hat{\mathbf{R}}(\alpha, \beta, \gamma) = \begin{pmatrix} 1 & -\gamma & \beta & t_x \\ \gamma & 1 & -\alpha & t_y \\ -\beta & \alpha & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The error metrics is defined as the sum of squares of perpendicular distances between corresponding points,

$$\hat{\mathbf{M}}_{\text{opt}} = \arg\min_{\hat{\mathbf{M}}} \sum_i \left(\left(\hat{\mathbf{M}} \cdot \mathbf{s}_i - \mathbf{d}_i \right) \bullet \mathbf{n}_i \right)^2.$$

Where,

s - Source point
d - Destination point
n - Normal at destination point

The error metrics can be rearranged to form a problem of the type Ax – b =0,

$$\min_{\hat{\mathbf{M}}} \sum_i \left(\left(\hat{\mathbf{M}} \cdot \mathbf{s}_i - \mathbf{d}_i \right) \bullet \mathbf{n}_i \right)^2 = \min_{\mathbf{x}} \left| \mathbf{A}\mathbf{x} - \mathbf{b} \right|^2.$$

These types of problems can be solved by Singular Value Decomposition (SVD) and Cholesky decomposition. The output is a transformation matrix M that maps the current depth frame to the previous one.

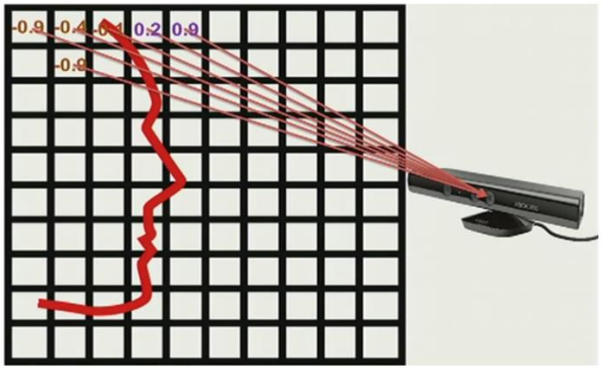
To limit processing time and increase efficiency, ICP is first performed on the coarsest level moving to the denser levels. The pose estimate from coarser levels are used as initial estimates for the next denser level.

3.3.6. Truncated Signed Distance Function (TSDF)

The final step involves a representation to store the final reconstruction model. The TSDF, a volumetric scene representation that allows for integration of multiple depth images taken from different viewpoints can be calculated as,

[TSDF value] = depth of pixel - [distance from sensor to voxel]

Get distance of the corresponding pixel of each voxel within the voxel grid. Subtract it from the distance of the voxel itself and divide by the truncation threshold.



The TSDF function has positive values at points x outside the surface, it decreases in value as x approaches the boundary of the surface where the signed distance function is zero, and it takes negative values inside of the reconstructed surface.

3.3.7. Divide depth frame into nodes

In order to capture the motion in a scene, the input depth frame is divided into smaller nodes. A volumetric *warp (motion) field* represents a set of sparse 6D transformation nodes that are smoothly interpolated through a k-nearest node average in the *canonical frame*.

Raw Depth Frame (Left) and frame divided into nodes (right).

3.3.8. ICP on individual nodes

Once the raw depth frame is divided into nodes, ICP is used to determine the transformation between current and previous node frames. We note that this step differs from *DynamicFusion's* technique of estimating the warp field. *DynamicFusion* formulates an energy function consisting of a dense model-to-frame ICP cost **Data** term, which is coupled with a regularization term **Reg**, that penalizes non-smooth motion fields, and ensures as-rigid-as-possible deformation between transformation nodes connected by edge set ε.

$$E(\mathcal{W}_t, \mathcal{V}, D_t, \mathcal{E}) = \mathbf{Data}(\mathcal{W}_t, \mathcal{V}, D_t) + \lambda \mathbf{Reg}(\mathcal{W}_t, \mathcal{E})$$

Where, $\mathcal{V}$, is the zero level set of TSDF, $\mathcal{W}_t$ represent the Warp field at time, D_t is the current raw depth frame and λ is the regularization constant.

We believe that for the purposes of this project, the use of **Data** term alone is sufficient to represent the dynamics of a scene. This can be justified by the fact that most of the chicken's body is usually stationary, with some motion observed in its head and tail.

3.3.9. Dual Quaternion Blending

Instead of transforming each node using only its own transformation, the transformation of its k-nearest nodes is averaged in order to maintain smoothness. This process is best implemented using Dual Quaternion Blending as described in [13]. The weighted average of unit dual quaternion is given by,

$$\mathbf{DQB}(x_c) \equiv \frac{\sum_{k \in N(x_c)} \mathbf{w}_k(x_c)\hat{\mathbf{q}}_{kc}}{\left\|\sum_{k \in N(x_c)} \mathbf{w}_k(x_c)\hat{\mathbf{q}}_{kc}\right\|},$$

Where, unit dual quaternion $\hat{\mathbf{q}}_{kc} \in \mathbb{R}^8$, N(x) are the k-nearest transformation nodes to the point x, w_k is the weight that determines the radius of influence of each node and ***SE3***(.) transforms converts quaternions back into an **SE(3)** transformation matrix.

3.3.10. Projective Signed Distance Function

The Truncated Signed Distance function utilized by KinectFusion is extended to operate over non-rigidly deforming scenes. Here we separately update the overall TSDF according to the deformation of each individual node. Thus, the final transformation that each point undergoes is a combination of the motion of the camera (Kinect) and the scene deformation. Once the TSDF is updated, the entire process is carried out on a newly acquired depth frame.

4. Results

All the models were created using depth data acquired from a Microsoft Kinect 2. The implementation was based on the open source implantation of KinectFusion, *Kinfu.* The reconstruction process is not real time and was implemented on Alienware machine with 1080 graphics card. In addition, our method does not require any pre-computation, and we do not rely on a pre-scanned template model – all reconstructions are built from scratch.

We show that ICP over smaller sections of a scene, can be used to produce a sufficiently detailed reconstruction for estimating weight of a chicken using its 3D model's volume. Our implementation of a linear algorithm that performs ICP over each node increases the run time significantly, thus behaving more like offline reconstruction techniques.

4.1. Parameters

The most important parameter is the number of nodes. There is a tradeoff better the resolution of the reconstruction and the run time for each iteration of the algorithm. While *DynamicFusion* uses about 400 nodes, we will limit our model to under 5 nodes. This is feasible because the goal requires a limited field of view for the reconstruction of a medium sized bird. Moreover, our implementation is limited by the GPU memory available to store each node's data separately. For the Dual Quaternion Blending, we use k = 4 nearest neighbors, similar to *DynamicFusion.*

4.2. Challenges

An exact implementation and a quantitative evaluation against DynamicFusion is challenging, since their method is hard to reproduce (their code is not publicly available and not all implementation details are given in the paper).

Another challenge encountered was the modification of *Kinfu* implementation to include the tracking of motions in a scene. Furthermore, since *Kinfu* is implemented using CUDA, debugging in parallel is difficult.

4.3. Limitations and Discussion

Due to limitation of time, our implementation does not take advantage of a GPU to parallelize the ICP step for each node is a frame. This project mainly serves as a proof of concept for showing that our technique can be used in poultry farms for estimating the growth rate of birds.

Furthermore, ICP assumes a very small movement between consecutive frames, therefore the reconstruction process will fail in case of fast movements, example a chicken running or turning too quickly. It is currently limited in its ability to achieve dynamic reconstruction of scenes that quickly move from a closed to open topology (for example starting a reconstruction with closed wings and then opening). More generally, failures common to real-time differential tracking can cause unrecoverable model corruption or result in loop closure failures. Large inter-frame motions, or motion of occluded regions, will also lead to an inaccurate surface prediction that prevents projective data-association in later frames. High levels of deformation, such as fully bending an object, may cause problems, as our regularizer distributes deformations smoothly over the grid.

Though there are several limitations to overcome, we believe that this project has far-reaching applications. On the production side our technique is beneficial in measuring conformity, animal growth rates and patterns, feeding habits (crop fill), size variabilities, and yield estimation. On the processing side, this technique can be implemented on products in motion on a processing plant and could provide new ways to perform sensing and grasping tasks with robotics. Moreover, due to the template-free nature of the algorithm, it could be able to be quickly migrated to modelling carcasses on processing lines and could be used as input into the next generation processing systems.

References

[1] Mortensen, Anders Krogh, Pavel Lisouski, and Peter Ahrendt. "Weight prediction of broiler chickens using 3D computer vision." *Computers and Electronics in Agriculture* 123 (2016): 319-326

[2] Usher, C. “Enhancing Predictive Modelling of Chickens”, Georgia Tech Research Institute.

[3] B. Brown and S. Rusinkiewicz. Non-Rigid Range-Scan Alignment Using Thin-Plate Splines. In Symposium on 3D Data Processing, Visualization, and Transmission, Sept. 2004.

[4] Izadi, Shahram, et al. "KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera." *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011.

[5] Newcombe, Richard A., Dieter Fox, and Steven M. Seitz. "Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[6] Innmann, M., Zollhöfer, M., Nießner, M. Theobalt, C., & Stamminger, M. "VolumeDeform: Real-time Volumetric Non-rigid Reconstruction." *arXiv preprint arXiv:1603.08161* (2016).

[7] W. Chang and M. Zwicker. Range Scan Registration Using Reduced Deformable Models. Comput. Graph. Forum, 28(2):447–456, 2009.

[8] Frost, A. R., Schofield, C. P., Beaulah, S. A., Mottram, T. T., Lines, J. A., & Wathes, C. M. "A review of livestock monitoring and the need for integrated systems." *Computers and Electronics in Agriculture* 17.2 (1997): 139-159.

[9] Schofield, C. P., Marchant, J. A., White, R. P., Brandl, N., & Wilson, M. "Monitoring pig growth using a prototype imaging system." *Journal of Agricultural Engineering Research* 72.3 (1999): 205-210.

[10] Wang, Y., Yang, W., Winter, P., & Walker, L. "Walk-through weighing of pigs using machine vision and an artificial neural network." *Biosystems Engineering* 100.1 (2008): 117-125.

[11] Mortensen, Anders Krogh, Pavel Lisouski, and Peter Ahrendt. "Weight prediction of broiler chickens using 3D computer vision." *Computers and Electronics in Agriculture* 123 (2016): 319-326.

[12] *Kinfu*, Point Cloud Library, Github repository (2014), https://github.com/PointCloudLibrary/pcl/tree/master/gpu/kinfu.

[13] L. Kavan, S. Collins, J. Zˇara, and C. O’Sullivan. Skinning ´ with Dual Quaternions. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D ’07, pages 39–46, New York, NY, USA, 2007. ACM.